



*Algoritmi

Noțiuni introductive

1.1. Noțiunea de algoritm.

Definiție: *Prin algoritm se înțelege o metodă de soluționare a unei clase de probleme, reprezentată de o succesiune finită de operații bine definite, numite instrucțiuni .*

Primul algoritm se considera algoritmul lui Euclid (utilizat pentru determinarea celui mai mare divizor comun a două numere naturale). Termenul de algoritm poate fi înțeles în sens larg nefiind neapărat legat de rezolvarea unei probleme cu caracter științific, ci doar pentru a descrie într-o manieră ordonată activități care constă în parcurgerea unei succesiuni de pași (cum este de exemplu utilizarea unui telefon public sau a unui bancomat).

În matematică există o serie de algoritmi: cel al rezolvării ecuației de gradul doi, algoritmul lui Eratostene (pentru generarea numerelor prime mai mici decât o anumită valoare), schema lui Horner (pentru determinarea câtului și restului împărțirii unui polinom la un binom) etc.

Soluția problemei se obține prin *execuția* algoritmului. Algoritmul poate fi executat pe o mașină formală (în faza de proiectare și analiză) sau pe o mașină fizică(calculator) după ce a fost codificat într-un limbaj de programare.

1.2. Caracteristicile unui algoritm

Generalitate. Un algoritm destinat rezolvării unei probleme trebuie să permită obținerea rezultatului pentru orice date de intrare și nu numai pentru date particulare de intrare.

Finitudine. Adică se termină după un număr finit de pași, indiferent cât de mulți.

Claritate. Prelucrările algoritmului trebuie specificate riguros, fără ambiguități. În orice etapă a execuției algoritmului trebuie să se știe exact care este următoarea etapă ce va fi executată.

Eficiență. Algoritmii pot fi efectiv utilizați doar dacă folosesc *resurse de calcul* în volum acceptabil.

Prin resurse de calcul se înțelege volumul de memorie și timpul necesar pentru execuție.

Exemple

1. *Nu orice problemă poate fi rezolvată algoritmic.*

a. Fiind dat un număr **n** să se determine toți divizorii săi.

Pentru această problemă se poate scrie un algoritm foarte ușor.

b. Fiind dat un număr **n** să se determine toți multipli săi.

Pentru această problemă **nu** se poate scrie un algoritm deoarece nu cunoaștem un criteriu de oprire a operațiilor.

2. *Un algoritm trebuie să funcționeze pentru orice date de intrare.*

Fiind date numerele **a**, **b**, **c** să se afișeze maximul dintre ele.

O posibilă soluție ar fi:

se compară **a** cu **b** și **c** și dacă e mai mare se afișează **a**, iar apoi

se compară **b** cu **a** și **c** și dacă e mai mare se afișează **b**, iar apoi

se compară **c** cu **b** și **a** și dacă e mai mare se afișează **c**

Algoritmul nu funcționează dacă 2 valori sunt identice și de valoare maximă.

Exemple

3. Un algoritm trebuie să se oprească.

Se consideră următoarea secvență de prelucrări:

Pas 1. Atribuire variabilei x valoarea 1;

Pas 2. Mărește valoarea lui x cu 2;

Pas 3. Dacă x este egal cu 100 atunci se oprește prelucrarea altfel se reia de la Pas 2.

Este ușor de observat că x nu va lua niciodată valoarea 100, deci succesiunea de prelucrări nu se termină niciodată. Din acest motiv nu poate fi considerată un algoritm corect.

4. Prelucrările dintr-un algoritm trebuie să fie neambigue.

Considerăm următoarea secvență de prelucrări:

Pas 1. Atribuire variabilei x valoarea 0;

Pas 2. *Fie* se mărește x cu 1 *fie* se micșorează x cu 1;

Pas 3. Dacă x aparține $[-10; 10]$ se reia de la Pas 2, altfel se oprește algoritmul.

Exemple

5. Un algoritm trebuie să se oprească după un interval rezonabil de timp.

Fiind dat un număr **n** se cere să se determine de câte ori a apărut o cifră **c** în reprezentarea tuturor numerelor naturale mai mici ca **n**.

O rezolvare simplă ar fi să luăm toate numere mai mici ca **n** și să vedem de câte ori apare cifra **c** în fiecare dinte ele. Soluția e simplă și pentru valori mici ale lui **n** algoritmul se termină într-un interval de timp rezonabil, dar pentru valori mari timpul de terminare al algoritmului crește nepermis de mult.

Pașii realizării unui algoritm

1. Citirea cu atenție a enunțului problemei.
2. Identificarea datelor de intrare și a celor de ieșire.
3. Rezolvarea propriu-zisă a problemei pe cazuri particulare și reprezentative.
În acest moment **nu** se încearcă scrierea programului ci doar determinarea metodei de rezolvare, generalizarea și înțelegerea acesteia.
4. Descrierea în limbaj natural a soluției problemei.
Dacă nu sunteți capabili să descrieți metoda folosită în limbaj natural e puțin probabil să o puteți face într-un limbaj de programare care e mai restrictiv decât limbajul natural.
5. Scrierea programului într-un limbaj de programare.
6. Testarea programului.
Testarea se face pe mai multe seturi de date care să acopere cazurile posibile ce pot apărea.

2.1. Date

Orice algoritm pornește de la anumite date de intrare, le prelucreză, iar în final obține date de ieșire .

Datele pot fi clasificate după tipul lor :

- * Numerice: · Întregi ; ex. 120, -120
· Reale ; ex. 3.12, 12.3
- * Logice; ex. TRUE(adevărat), FALSE(fals)
- * Șir de caractere; ex. ‘un text’.

2.2. Variabile

Tipurile de variabile coincid cu tipurile de date.
Printr-o variabilă înțelegem un ansamblu de patru elemente:

- * Numele variabilei ;
- * Tipul ei ;
- * Valoarea ei la un moment dat ;
- * Locul în memoria calculatorului unde poate fi găsită variabila (adresa ei)

2.3. Expresii

Cu ajutorul constantelor, variabilelor si operatorilor se pot construi diverse expresii. În scrierea expresiilor este permisă folosirea parantezelor.

Expresiile sunt de mai multe tipuri :

- * Întregi ;
- * Reale ;
- * Logice ;
- * De tip şir de caractere.

Expresii întregi

Folosesc operatorii +, -, *, DIV (pentru câtul împărțirii întregi), MOD (pentru restul împărțirii întregi)s.a.

Rezultatul evaluării lor este un număr întreg .

Exemple :

- * $3*x+7$, unde x este variabila întreagă ;
- * $a*(b+c)$, unde a , b , c sunt variabile întregi .

Expresii reale

Operanții care le alcătuiesc sunt constante și variabile întregi sau reale. Pe lângă operatorii utilizați în cadrul expresiilor întregi putem utiliza și alții cum ar fi de exemplu: '/' (pentru împărțirea cu zecimale). Rezultatul evaluării lor este întotdeauna un număr real.

Expresii logice

Rezultatul lor poate avea numai două valori: TRUE sau FALSE .

Operatorii sunt cei din logica matematică : OR, AND, NOT . De asemenea este permis să folosim și operatorii =, <, >, >= (mai mare sau egal), <= (mai mic sau egal), <> (diferit).

Exemple :

a OR b, unde **a** și **b** sunt variabile logice ;
(a>=b) AND (c<d) .

Expresii de tip șir de caractere

Se folosesc constante și variabile de tip șir de caractere. Singurul operator permis este operatorul + având semnificația de concatenare (scrierea a celor două șiruri unul după altul).

Exemplu :

`a + 'text'`, unde `a` este o variabilă de tip șir de caractere având conținutul 'acest ' are ca rezultat șirul 'acest text'.

2.4. Operații

Clasificare operații :

- * Operații de intrare și ieșire ;
- * Operații de atribuire ;
- * Operații de decizie ;

Operații de intrare și ieșire

Prin operația de intrare (de citire) se înțelege preluarea unei date de la un dispozitiv de intrare (tastatură, unitatea de dischetă, unitatea de disc) către memoria internă a calculatorului, în spațiul rezervat pentru aceasta (spațiul rezervat pentru variabila care primește ca valoare acea dată).

Prin operația de ieșire (scriere) se înțelege trecerea unei date din memoria internă către un dispozitiv de ieșire (monitorul, unitatea de dischetă, unitatea de disc).

Operații de atribuire

Să presupunem că un algoritm folosește o variabilă notată arbitrar z și o alta notată y . Dorim ca z să ia o anumită valoare (de ex. 2). Spunem că am atribuit variabilei z valoarea 2 și pentru moment vom nota acest lucru astfel: $z:=2$. Atât timp cât asupra variabilei z nu efectuăm nici o altă operație de atribuire și nu efectuăm nici o operație de citire, aceasta își păstrează valoarea 2. Presupunem că scriem valoarea variabilei z pe monitor. După scriere variabila z va avea în continuare valoarea 2. Presupunem că atribuim variabilei y valoarea variabilei z ($y:=z$). În urma atribuirii, valoarea variabilei y va fi 2 și valoarea variabilei z va rămâne 2. Dacă atribuim variabile z valoarea $z+z+7$ ($z:=z+z+7$) atunci z va avea valoarea 11.

Operația de atribuire se efectuează astfel :

- * Se evaluează expresia din partea dreaptă a operației de atribuire ;
- * Valoarea care se obține astfel este preluată de variabila din partea stângă, iar valoarea avută de aceasta se pierde.

Operații de decizie

În general, în funcție de anumite condiții, trebuie făcute anumite operații sau altele. Operația prin care testăm acele condiții se numește operația de decizie. În funcție de rezultatul testului, algoritmul execută anumite operații.

1.3. Structuri de bază (liniară, alternativă, repetitivă)

Programarea structurată este o metodă independentă de limbajul de programare, ea acționând la nivelul stilului de lucru. Programarea structurată reprezintă o manieră de concepere a programelor potrivit unor reguli bine stabilite, utilizând un anumit set, redus, de *tipuri de structuri de control* .

Programarea structurată poate fi reprezentată ca o combinație a trei structuri de control :

- * Secvența (succesiunea de două sau mai multe operații) ;
- * Decizia (alegerea unei operații dintre două alternative posibile) ;
- * Ciclul cu test inițial (repetarea unei operații atâta timp cât o anumită condiție este îndeplinită) .

Programarea structurată admite și utilizarea altor structuri de control, cum sunt :

- * Selecția (permite o alegere între mai mult de două alternative) ;
- * Ciclul cu test final ;
- * Ciclul cu contor .

Realizatori:

- *Iliescu Vlad*
- *Nicula George*